

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800

Open access books available

122,000

International authors and editors

135M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



A Simple Hybrid Particle Swarm Optimization

Wei-Chang Yeh

*Department of Industrial Engineering and Engineering Management
National Tsing Hua University
Taiwan, R.O.C.*

1. Introduction

As a novel stochastic optimization technique, the Particle Swarm Optimization technique (PSO) has gained much attention towards several applications during the past decade for solving the global optimization problem or to set up a good approximate solution to the given problem with a high probability. PSO was first introduced by Eberhart and Kennedy [Kennedy and Eberhart, 1997]. It belongs to the category of Swarm Intelligence methods inspired by the metaphor of social interaction and communication such as bird flocking and fish schooling. It is also associated with wide categories of evolutionary algorithms through individual improvement along with population cooperation and competition. Since PSO was first introduced to optimize various continuous nonlinear functions, it has been successfully applied to a wide range of applications owing to the inherent simplicity of the concept, easy implementation and quick convergence [Trelea 2003].

PSO is initialized with a population of random solutions. Each individual is assigned with a randomized velocity based to its own and the companions flying experiences, and the individuals, called particles, are then flown through hyperspace. PSO leads to an effective combination of partial solutions in other particles and speedens the search procedure at an early stage in the generation. To apply PSO, several parameters including the population (N), cognition learning factor (c_p), social learning factor (c_g), inertia weight (w), and the number of iterations (T) or CPU time should be properly determined. Updating the velocity and positions are the most important parts of PSO as they play a vital role in exchanging information among particles. The details will be given in the following sections.

The simple PSO often suffers from the problem of being trapped in local optima. So, in this paper, the PSO is revised with a simple adaptive inertia weight factor, proposed to efficiently control the global search and convergence to the global best solution. Moreover, a local search method is incorporated into PSO to construct a hybrid PSO (HPSO), where the parallel population-based evolutionary searching ability of PSO and local searching behavior are reasonably combined. Simulation results and comparisons demonstrate the effectiveness and efficiency of the proposed HPSO.

The paper is organized as follows. Section 2 describes the acronyms and notations. Section 3 outlines the proposed method in detail. In Section 4, the methodology of the proposed HPSO is discussed. Numerical simulations and comparisons are provided in Section 5. Finally, Concluding remarks and directions for future work are given in in Section 6.

Source: Advances in Evolutionary Algorithms, Book edited by: Witold Kosiński, ISBN 978-953-7619-11-4, pp. 468, November 2008, I-Tech Education and Publishing, Vienna, Austria

2. Acronym and notations

Acronym:

PSO : Particle Swarm Optimization Algorithm

SPSO : Traditional PSO

IPSO : An improved PSO proposed in [Jiang *et. al.* 2007]

HPSO : The proposed Hybrid PSO

Notations:

D : The number of dimensions.

N : The number of particles in each replication.

T : The number of generations in each replication.

R : The total number of independent replications.

r_{\bullet} : The random number uniformly distributed in [0, 1].

c_p, c_g : The cognition learning factor and the social learning factor, respectively.

w : The inertia weight.

$x_{t,i,j}$: The dimension of the position of particle i at iteration t, where $t=1,2,\dots,T$, $i=1,2,\dots,N$, and $j=1,2,\dots,D$.

$X_{t,i}$: $X_{t,i}=(x_{t,i,1},\dots,x_{t,i,D})$ is the position of particle i at iteration t, where $t=1,2,\dots,T$, $i=1,2,\dots,N$, and $j=1,2,\dots,D$.

$v_{t,i,j}$: the dimension of the velocity of particle i at iteration t, where $t=1,2,\dots,T$, $i=1,2,\dots,N$, and $j=1,2,\dots,D$.

$V_{t,i}$: $V_{t,i}=(v_{t,i,1},\dots,v_{t,i,D})$ is the velocity of particle i at iteration t, where $t=1,2,\dots,T$, $i=1,2,\dots,N$, and $j=1,2,\dots,D$.

$P_{t,i}$: $P_{t,i}=(p_{t,i,1},\dots,p_{t,i,D})$ is the best solution of particle i so far until iteration t, i.e., the pBest, where $t=1,2,\dots,T$, $i=1,2,\dots,N$, and $j=1,2,\dots,D$.

G_t : $G_t=(g_{t,1},\dots,g_{t,D})$ the best solution among $P_{t,1}, P_{t,2}, \dots, P_{t,N}$ at iteration t, i.e., the gBest, where $t=1,2,\dots,T$.

$F(\bullet)$: The fitness function value of \bullet .

$U(\bullet), L(\bullet)$: The upper and lower bounds for \bullet , respectively.

3. The PSO

In PSO, a solution is encoded as a finite-length string called a particle. All of the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles [Parsopoulos *et. al.* 2001]. PSO is initialized with a population of random particles with random positions and velocities inside the problem space, and then searches for optima by updating generations. It combines the local and global search resulting in high search efficiency. Each particle moves towards its best previous position and towards the best particle in the whole swarm in every iteration. The former is a local best and its value is called *pBest*, and the latter is a global best and its value is called *gBest* in the literature. After finding the two best values, the particle updates its velocity and position with the following equation in continuous PSO:

$$v_{t,i,j} = wv_{t-1,i,j} + c_p r_{p,i,j} (p_{t-1,i,j} - x_{t-1,i,j}) + c_g r_{g,i,j} (g_{t-1,j} - x_{t-1,i,j}), \quad (1)$$

$$x_{t,i,j} = x_{t-1,i,j} + v_{t,i,j}. \quad (2)$$

The values $c_p r_{p,ij}$ and $c_g r_{g,ij}$ determine the weights of the two parts, and $c_p + c_g$ is usually limited to 4 [Trelea 2003]. Generally, the value of each component in $X_{t,i}$ and $V_{t,i}$ can be clamped to the range $[L(X), U(X)]$ and $[L(V), U(V)]$, respectively, to limit each particle to ensure its feasibility.

For example, let

$$X_{3,4} = (1.5, 3.6, 3.7, -3.4, -1.9), \quad (3)$$

$$V_{2,4} = (0.4, 0.1, 0.7, -2.7, -3.5), \quad (4)$$

$$P_{3,4} = (1.6, 3.7, 3.5, -2.1, -1.9), \quad (5)$$

$$G_3 = (1.7, 3.7, 2.2, -3.5, -2.5), \quad (6)$$

$$R_p = (0.21, 0.58, 0.73, 0.9, 0.16), \quad (7)$$

$$R_g = (0.47, 0.45, 0.28, 0.05, 0.77), \quad (8)$$

$$L(X) = (0, 0, 0, -3.6, -3), \quad (9)$$

$$U(X) = (2, 4, 4, 0, 0), \quad (10)$$

$$L(V) = (-4, -4, -4, -4, -4), \quad (11)$$

$$U(V) = (4, 4, 4, 4, 4), \quad (12)$$

$$w = 0.9, \quad (13)$$

$$c_p = c_g = 2. \quad (14)$$

Then, from Eq.(1), we have

$$V_{3,4} = (0.59, 0.296, -0.502, -0.1, -4.074). \quad (15)$$

Since $-4.074 < -4$, $V_{3,4}$ needs to be adjustment in the following:

$$V_{3,4} = (0.59, 0.296, -0.502, -0.1, -4). \quad (16)$$

Under the guidance of Eq.(2),

$$X_{4,4} = (2.09, 3.896, 3.198, -3.5, -5.974), \quad (17)$$

and

$$X_{4,4} = (2.0, 3.896, 3.198, -3.5, -3.0) \quad (18)$$

after the adjustment according to the upper/lower-bounds of X .

We conducted the preliminary experiments, and the complete computational procedure of the PSO algorithm can be summarized as follows.

STEP 1: Initialize: Initialize parameters and population with random positions and velocities.

- STEP 2: Evaluation: Evaluate the fitness value (the desired objective function) for each particle.
- STEP 3: Find the *pBest*: If the fitness value of particle *i* is better than its best fitness value (*pBest*) in history, then set current fitness value as the new *pBest* to particle *i*.
- STEP 4: Find the *gBest*: If any *pBest* is updated and is better than the current *gBest*, then set *gBest* to the current value.
- STEP 5: Update and adjustment velocity: Update velocity according to Eq.(1). Adjust the velocity to meet its range if necessary.
- STEP 6: Update and adjustment position: Update velocity and move to the next position according to Eq.(2). Adjust the position to meet their range if necessary.
- STEP 7: Stopping criterion: If the number of iterations or CPU time are met, then stop; otherwise go back to STEP 2.

4. The proposed HPSO

To overcome the weakness of PSO for local searches, this paper aims at creating HPSO by combining PSO, local search (LS), and vector based (VB) with a linearly varying inertia weight. The PSO part in the proposed HPSO is similar to the SPSO proposed in section 3. Hence, only the differences, i.e. the linearly varying inertia weight, LS and VB, are elaborated in this section.

4.1 Initial population

The initial population is generated randomly in the feasible space such that its lower-/upper-bounds are satisfied. To construct a direct relationship between the problem domain and the PSO particles in this study, the *i*th dimension in the particle stands for the value of the *i*th variable in the solution.

4.2 The linearly varying inertia weight

One of the most important issues to find the optimum solution effectively and efficiently while designing the PSO algorithm is its parameters. The inertia weight represents the influence of previous velocity which provides the necessary momentum for particles to move across the search space. Hence, the inertia weight dictates the balance between exploration and exploitation in PSO [Jiang *et. al.* 2007]. Shi and Eberhart (2001) made a significant improvement in the performance of the PSO with a linearly varying inertia weight over the generations, which linearly varies from 0.9 at the beginning of the search to 0.4 at the end. Thus the linearly varying inertia weight is adapted in the proposed HPSO to achieve trade-off between exploration and exploitation, i.e. the inertia weight of the *i*th generation is

$$w_i = U(w) - (i-1)[U(w) - L(w)]/N. \quad (19)$$

4.3 Vector based PSO

The underlying principle of the traditional PSO is that the next position of each particle is a compromise of its current position, the best position in its history so far, and the best position among all existing particles. The vector synthesis is the original mathematical foundation of PSO, as shown in the following figure.

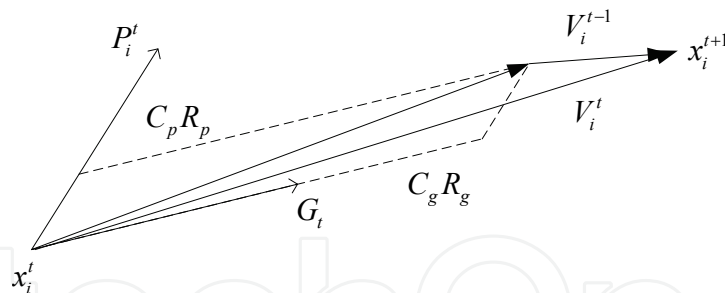


Fig. 1. The vector synthesis of PSO.

Eqs.(1) and (2) are more complicated than the concept of the vector synthesis in increasing the diversity of the dimensions of each particle. Hence, the following equations are implemented in the proposed HPSO instead of Eqs.(1) and (2):

$$V_{t,i} = w_i V_{t-1,i} + c_p r_p (P_{t-1,i} - X_{t-1,i}) + c_g r_g (G_{t-1} - X_{t-1,i}), \quad (20)$$

$$X_{t,i} = X_{t-1,i} + V_{t,i}. \quad (21)$$

In the traditional PSO, each particle needs to use two random number vectors (e.g., R_p and R_q) to move to its next position. However, only two random number (e.g., r_p and r_q) are needed in the proposed HPSO. Besides, Eqs(20) and (21) are very easy and efficient in deciding the next positions for the problems with continuous variables. For example, let $P_{3,4}$, $X_{3,4}$, $P_{3,4}$, G_3 , $L(X)$, $U(X)$ are the same as defined in the example in Section 1. Assume $r_p=0.34$ and $r_g=0.79$. From Eq.(19),

$$w_i = 0.9 - (4-1)[0.9-0.4]/1000 = 0.8985. \quad (22)$$

Plug w_i , r_p , r_g and the other required value into Eq.(20), we have

$$V_{3,4} = (0.6974, 0.28985, -1.56505, -1.75795, -3.97275), \quad (23)$$

$$V_{3,4} = (0.6974, 0.28985, -1.56505, -1.75795, -3.97275), \quad (24)$$

where $X_{4,4}$ is adjustment from

$$(2.1974, 3.88985, 2.13495, -5.15795, -5.87275). \quad (25)$$

4.4 Local search method

One of the major drawbacks of PSO is its very slow convergence. To surmount this drawback, to guide the search towards unexplored regions in the solution space and to avoid being trapped into local optimum, LS is implemented for constructing the proposed HPSO.

In PSO, proper control of global exploration and local exploitation is crucial in finding the optimum solution efficiently [Liu 2005]. A local optimizer is applied to the best particle (i.e. $gBest$) for each run in order to push it to climb the local optimum [Liu 2005]. With the hybrid method, PSOs are used to perform global exploration around particles except the $gBest$ to maintain population diversity, while the local optimizer is used to perform local exploitation to the best particle. Since the properties of PSOs and conventional local optimizers are complementary, HPSOs are often better than either method operating alone from the computation experiments shown in Section 5.

The proposed LS is very simple and similar to the famous local improvement method the pairwise exchange procedure. In LS, the i th dimension of both the current best particle of all population (i.e., $gBest$) are replaced by the current best particle of the j th particle (i.e., $pBest$). If the fitness function value is improved, the the current $gBest$ is updated accordingly. Otherwise, there is no need to change the current $gBest$. The above procedure in the proposed HPSO is repeated until all dimensions in the $gBest$ are performed.

To minimize the number of duplicated computations of the same fitness function in LS, only one non- $gBest$ is randomly selected to each dimension of $gBest$ in the local search. The complete procedure of the local search part of the proposed HPSO can be summarized as in the following:

STEP 0. Let $d=1$.

STEP 1. Let $n=1$.

STEP 2. If $G_d=P_{t,n}$ or $g_{t,d}=p_{t,n,d}$, go to STEP 4. Otherwise, let $F^*=F(G_d)$, $G_d=P_{t,n}$ and $g_{t,d}=p_{t,n,d}$.

STEP 3. If $F(G_d)$ is better than F^* , then let $F^*=F(G_d)$. Otherwise, let $g_{t,d}=g$.

STEP 4. If $n<N$, let $n=n+1$ and go to STEP 2.

STEP 5. If $d<D$, let $d=d+1$ and go to STEP 1.

5. Numerical examples

To evaluate the performance of the proposed algorithms, four famous benchmark optimization problems [Jiang *et. al.* 2007] are used, which are described as follows.

Function	Formula	Range	Optima	Solution
Rosenbrock	$f_1(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	$[-30,30]^n$	0	$(1,...,1)$
Rastrigrin	$f_2(x) = \sum_{i=1}^n \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	$[-5.12,5.12]^n$	0	$(0,...,0)$
Griewark	$f_3(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600,600]^n$	0	$(0,...,0)$

Table 1. Benchmark functions.

Features of the above three functions are the following: Rosenbrock is an unimodal function and its variables are strongly dependent and gradient information often misleads algorithms; Rastrigrin is ultimodal function with many local optima; Griewank is strongly multi-modal with significant interactions between its variables (caused by the product term) and the number of local minima increases with dimensionality [Jiang *et. al.* 2007].

These problems are implemented using the proposed HPSO, SPSO, and the best-known PSO (IPSO) proposed in by Jiang *et. al.* (2007) with regard to these three benchmark problems and the results of the experiments were compared. The proposed HPSO, SPSO and IPSO were implemented in C programming language on an Intel Pentium 2.6 GHz PC with 2G memory. To facilitate fair comparison, the number of iterations, the number of runs, and the populations for the proposed HPSO, SPSO and IPSO are taken directly from Jiang *et. al.* (2007). All these methods use a linearly varying inertia weight over the generations, varying from 0.9 at the beginning of the search to 0.4 at the end. In addition, $c_g=c_p=2$, $X_{max}=V_{max}=UB$ and $X_{min}=V_{min}=LB$ are used.

As in Jiang *et. al.* (2007), the proposed HPSO, SPSO and IPSO were tested on four group problems (population sizes of 20, 40, 80, and 160). The population sizes of each group are equal and each group problem contains 3 data sets. Each data set was first run with 3 sets of dimensions: 10, 20, and 30 and the corresponding maximum number of generations are set

as 1000, 1500 and 2000, respectively. Hence, there are 12 different test sets to each benchmark problem as follows:

- Set A1: $N=20, D=10, T=1000$;
- Set A2: $N=20, D=20, T=1500$;
- Set A3: $N=20, D=30, T=2000$;
- Set B1: $N=40, D=10, T=1000$;
- Set B2: $N=40, D=20, T=1500$;
- Set B3: $N=40, D=30, T=2000$;
- Set C1: $N=80, D=10, T=1000$;
- Set C2: $N=80, D=20, T=1500$;
- Set C3: $N=80, D=30, T=2000$;
- Set D1: $N=160, D=10, T=1000$;
- Set D2: $N=160, D=20, T=1500$;
- Set D3: $N=160, D=30, T=2000$;

Each algorithm with each set of parameter is executed in 50 independent runs. The average fitness values of the best particle found for the 50 runs for the three functions are listed in Table 2. The shaded number shows the best result with respect to the corresponding function and the set. From the Table 2, it can be seen that IPSO outperforms the HPSO in Rosenbrock functions for POP=20 and 80. However, the remaining cases of Rosenbrock functions and for the Griewark function, the proposed HPSO has almost achieved better results than SPSO and IPSO. Furthermore HPSO is superior to SPSO and IPSO at all instances of the Rastrigrin function.

SET	Rosenbrock			Rastrigrin			Griewark		
	PSO	IPSO	HPSO	PSO	IPSO	HPSO	PSO	IPSO	HPSO
A1	42.6162	10.5172	3.3025	5.2062	3.2928	0	0.0920	0.0784	0.0071
A2	87.2870	75.7246	124.3305	22.7724	16.4137	0.4975	0.0317	0.0236	0.0168
A3	132.5973	99.8039	122.7829	49.2942	35.0189	1.0760	0.0482	0.0165	0.0190
B1	24.3512	1.2446	0	3.5697	2.6162	0	0.0762	0.0648	0.0002
B2	47.7243	8.7328	0.0797	17.2975	14.8894	0	0.0227	0.0182	0.0026
B3	66.6341	14.7301	120.7434	38.9142	27.7637	0	0.0153	0.0151	0.0012
C1	15.3883	0.1922	0.0797	2.3835	1.7054	0	0.0658	0.0594	0
C2	40.6403	1.5824	60.3717	12.9020	7.6689	0	0.0222	0.0091	0
C3	63.4453	1.5364	4.7461	30.0375	13.8827	0	0.0121	0.0004	0
D1	11.6283	0.0598	0	1.4418	0.8001	0	0.0577	0.0507	0
D2	28.9142	0.4771	0	10.0438	4.2799	0	0.0215	0.0048	0
D3	56.6689	0.4491	0	24.5105	11.9521	0	0.0121	0.0010	0
Average	39.48832	3.22272	20.66896	15.67783	9.50649	0	0.03396	0.02483	0.00044

Table 2. Mean Fitness function values 50 independent runs.

The final statistical result including the Success Rate, the fitness function values, CPU times and convergence iterations of all 50 runs related to Rosenbrock function, Rastrigrin function, and Griewark function are listed in Tables 3-5. The Success Rate is defined to be the percentage of the number of final searching solution that is equal to the global optimal value in 50 independent runs. Convergence iterations denote the number of iterations required for convergence. These data are divided into three categories: maximum, minimum, average, and standard deviations (denoted by max, min, mean, and std., respectively).

SET	Success	Fitness Function Value				Running Time (sec.)				Convergence Iterations			
	Rate	max	min	mean	std	max	min	Mean	std	max	min	mean	std
A1	92%	153.17	0	3.3025	21.65	0.03	0.02	0.02	0.00	962	520	720.3	73.77
A2	90%	3018.59	0	124.3305	597.18	0.12	0.05	0.06	0.01	1264	520	1130.3	141.50
A3	82%	3018.59	0	122.7829	597.14	0.29	0.09	0.11	0.04	1987	914	1603.5	268.45
B1	100%	0	0	0	0	0.05	0.04	0.05	0.00	787	607	660.1	39.34
B2	98%	3.99	0	0.0797	0.56	0.25	0.09	0.14	0.03	1129	813	1024.7	64.66
B3	96%	3018.59	0	120.7434	597.52	0.47	0.19	0.23	0.07	1635	1176	1477.9	114.45
C1	98%	3.99	0	0.0797	0.56	0.13	0.07	0.11	0.01	782	556	602.7	37.37
C2	98%	3018.59	0	60.3717	426.89	0.46	0.24	0.37	0.04	1089	649	946.9	60.43
C3	98%	237.31	0	4.7461	33.56	0.91	0.44	0.66	0.10	1457	1217	1346.7	56.36
D1	100%	0	0	0	0	0.27	0.21	0.25	0.02	681	533	567.4	28.35
D2	100%	0	0	0	0	1.00	0.78	0.89	0.05	1048	818	887.3	47.31
D3	100%	0	0	0	0	2.41	1.52	1.78	0.17	1452	1073	1232.0	73.77

Table 3. Experimental results on Rosenbrock function of 50 independent runs.

SET	Success	Fitness Function Value				Running Time (sec.)				Convergence Iterations			
	Rate	max	min	mean	std	max	min	mean	std	max	min	mean	std
A1	100%	0	0	0	0	0.05	0.04	0.05	0.00	112	5	30.3	20.52
A2	98%	24.87	0	0.4975	3.52	0.16	0.15	0.16	0.00	329	11	56.1	49.95
A3	96%	28.92	0	1.0760	5.34	0.37	0.33	0.34	0.01	357	19	86.4	74.91
B1	100%	0	0	0	0	0.09	0.08	0.08	0.00	58	7	25.4	12.34
B2	100%	0	0	0	0	0.27	0.25	0.25	0.00	104	11	36.4	20.96
B3	100%	0	0	0	0	0.58	0.53	0.55	0.01	196	20	47.7	28.55
C1	100%	0	0	0	0	0.17	0.15	0.16	0.00	56	5	19.3	11.02
C2	100%	0	0	0	0	0.49	0.45	0.47	0.01	106	9	32.9	17.43
C3	100%	0	0	0	0	1.01	0.92	0.96	0.02	127	18	45.2	25.74
D1	100%	0	0	0	0	0.32	0.30	0.31	0.00	38	5	15.6	7.43
D2	100%	0	0	0	0	0.92	0.86	0.89	0.01	63	9	26.4	10.87
D3	100%	0	0	0	0	1.91	1.74	1.79	0.03	97	9	34.7	15.96

Table 4. Experimental results on Rastrigrin function of 50 independent runs.

SET	Success	Fitness Function Value				Running Time (sec.)				Convergence Iterations			
	Rate	max	min	mean	std	max	min	mean	std	max	min	mean	std
A1	86%	0.10	0	0.0071	0.02	0.08	0.08	0.08	0.00	711	6	110.7	169.22
A2	74%	0.13	0	0.0186	0.04	0.30	0.27	0.28	0.01	1401	18	262.1	303.67
A3	76%	0.21	0	0.0190	0.04	0.72	0.64	0.67	0.03	812	15	275.0	272.40
B1	98%	0.01	0	0.0002	0.00	0.14	0.13	0.14	0.00	518	4	53.1	74.79
B2	96%	0.07	0	0.0026	0.01	0.49	0.44	0.45	0.01	823	10	90.7	151.76
B3	96%	0.03	0	0.0012	0.01	1.06	0.99	1.00	0.01	738	10	95.7	140.38
C1	100%	0	0	0	0	0.26	0.25	0.26	0.00	124	2	20.5	23.07
C2	100%	0	0	0	0	0.81	0.78	0.80	0.01	120	10	40.8	26.87
C3	100%	0	0	0	0	1.74	1.67	1.70	0.02	131	8	48.1	28.07
D1	100%	0	0	0	0	0.51	0.49	0.50	0.00	30	6	14.8	6.09
D2	100%	0	0	0	0	1.53	1.46	1.49	0.01	104	9	29.0	16.99
D3	100%	0	0	0	0	3.18	3.04	3.10	0.03	77	9	32.9	14.44

Table 5. Experimental results on Griewark function of 50 independent runs.

As the dimension increases, the solution space get more complex, and PSO algorithm is more likely to be trapped into local optima. Experimental data shown in Table 2 does not clearly indicate that the HPSO outperforms the other PSOs in the measures of average fitness function values. However, the Success Rates are all over 74%. Therefore, the proposed HPSO can find global optima with very high probability, and it is concluded that HPSO has the strongest exploration ability and it is not easy to be trapped into local optima. Table 3 shows that the proposed HPSO uses only 3.18 seconds in worst case and 0.6 seconds in average. Thus, HPSO is very effective, efficient, robust, and reliable for complex numerical optimization.

6. Conclusions

A successful evolutionary algorithm is one with a proper balance between exploration (searching for good solutions), and exploitation (refining the solutions by combining information gathered during the exploration phase). In this study, a new hybrid version of PSO called HPSO is proposed. The HPSO constitutes a vector based PSO method with the linearly varying inertia weight, along with a local search.

A novel, simpler, and efficient mechanism is employed to move the $gBest$ to its next position in the proposed HPSO. The HPSO combines the population-based evolutionary searching ability of PSO and local searching behavior to efficiently balance the exploration and exploitation abilities. The result obtained by HPSO has been compared with those obtained from traditional simple PSO (SPSO) and improved PSO (IPSO) proposed recently. Computational results show that the proposed HPSO shows an enhancement in searching efficiency and improve the searching quality. In summary, the results presented in this work are encouraging and promising for the application of the proposed HPSO to other complex problems.

Further analysis is necessary to see how other soft computing method (e.g., the genetic algorithm, the taboo search, etc.) react to local searches for future researchers who may want to develop their own heuristics and to make further improvements. Our research is still very active and under progress, and it opens the avenues for future efforts in this directions such as: how to adjust parameters, increase success rates, reduce running times, using other local search, and the aggregation of different and new concepts to PSO.

7. References

- B. Liu, L. Wang, Y.-H. Jin, F. Tang, D.-X. Huang (2005), "Improved particle swarm optimization combined with chaos", *Chaos Solitons & Fractals*, Vol. 25, 2005, pp. 1261-1271.
- I.C. Trelea (2003), The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters*, Vol. 85, 2003, pp. 317-325.
- J. Kennedy and R.C. Eberhard (1995), "Particle swarm optimization", *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, USA, 1995, pp. 1942-1948.
- J. Kennedy and R.C. Eberhard and Y. Shi, "Swarm intelligence", San Francisco, CA: Morgan Kaufmann; 2001.
- J. Kennedy and R.C. Eberhart (1997), "A discrete binary version of the particle swarm algorithm", *Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, IEEE International Conference*, Vol. 5, No. 12-15, 1997/10, pp. 4104-4108.

- J. Moore and R. Chapman (1999), "Application of particle swarm to multiobjective optimization", *Department of Computer Science and Software Engineering, Auburn University*.
- K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis (2001), Improving particle swarm optimizer by function stretching, *Advances in Convex Analysis and Global Optimization*, 2001, 445–457.
- R.C. Eberhart and Y. Shi (2001), "Particle Swarm Optimization: Developments, Application and Resources", *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, South Korea, Vol. 1, pp. 81-86.
- Y. Jiang, T. Hu, C. Huang, and X. Wu (2007), "An improved particle swarm optimization algorithm", *Applied Mathematics and Computation*, Vol. 193, pp. 231–239.

IntechOpen



Advances in Evolutionary Algorithms

Edited by Xiong Zhihui

ISBN 978-953-7619-11-4

Hard cover, 284 pages

Publisher InTech

Published online 01, November, 2008

Published in print edition November, 2008

With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice. Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Wei-Chang Yeh (2008). A Simple Hybrid Particle Swarm Optimization, Advances in Evolutionary Algorithms, Xiong Zhihui (Ed.), ISBN: 978-953-7619-11-4, InTech, Available from:
http://www.intechopen.com/books/advances_in_evolutionary_algorithms/a_simple_hybrid_particle_swarm_optimization

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen